An introduction to reinforcement learning

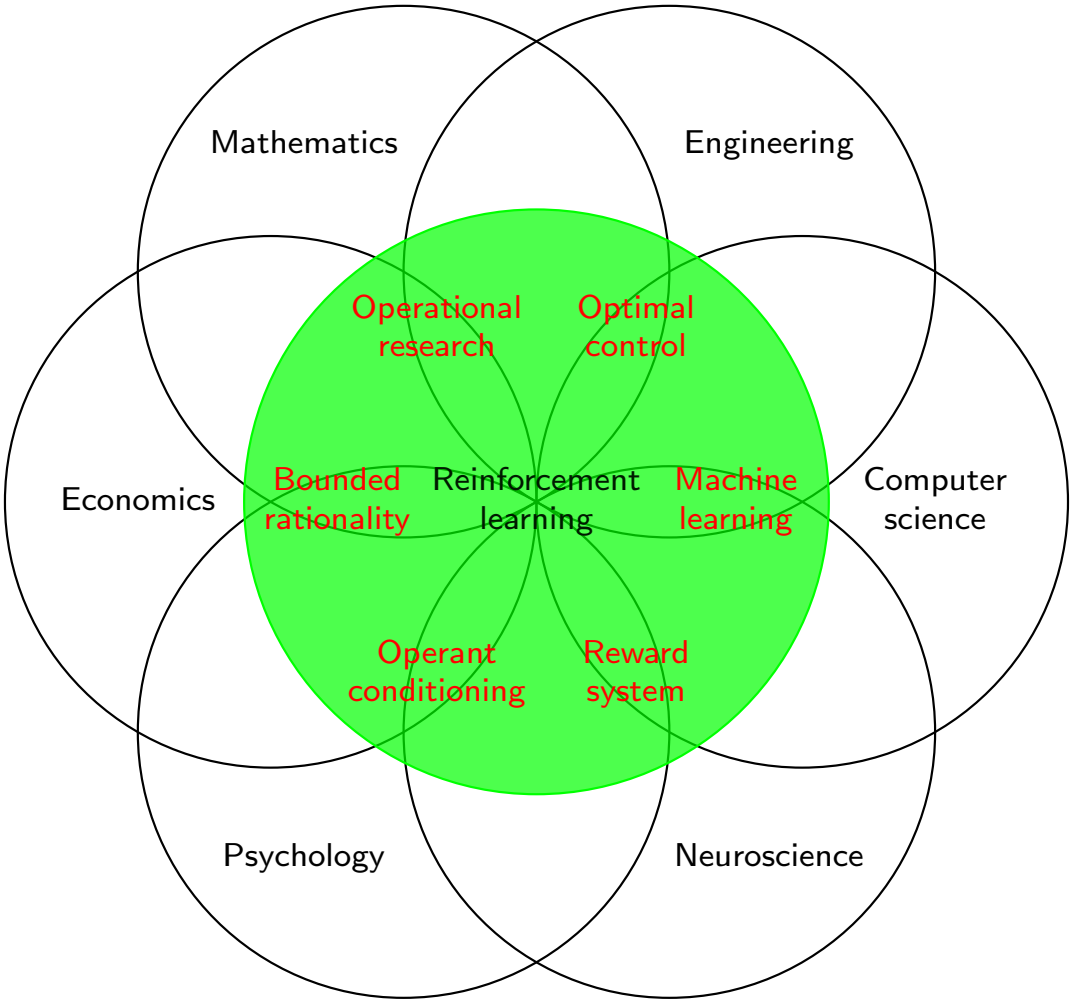
challenges and opportuinities

Corrado Possieri

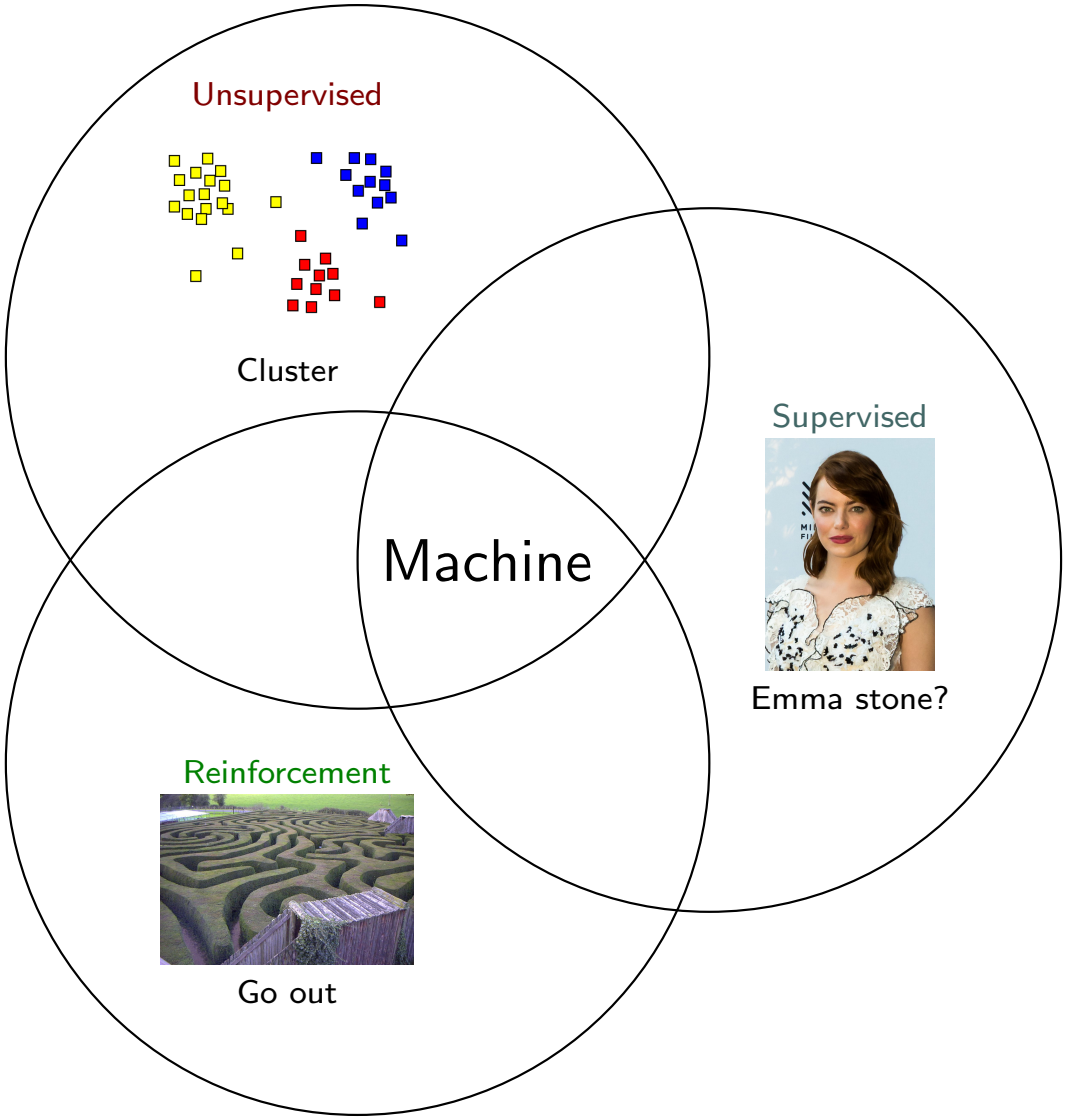
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti",
Consiglio Nazionale delle Ricerche, 00185 Roma, Italy

SMART-SEMINARS@IASI
July 30, 2020

Many aspects of reinforcement learning



Many faces of learning



What is reinforcement learning

Unique characteristics of reinforcement learning:

- *no supervisor*, just a reward signal;
- feedback is delayed;
- no need of correcting sub-optimal actions;
- data are not i.i.d., that is *time matters*;
- data depend on agent's actions.

Examples

- Autonomous driving.
- Play chess.
- Manage investment portfolio.
- Learn to walk.
- Control a power station.
- Play games.
- Minimize epidemic spread.

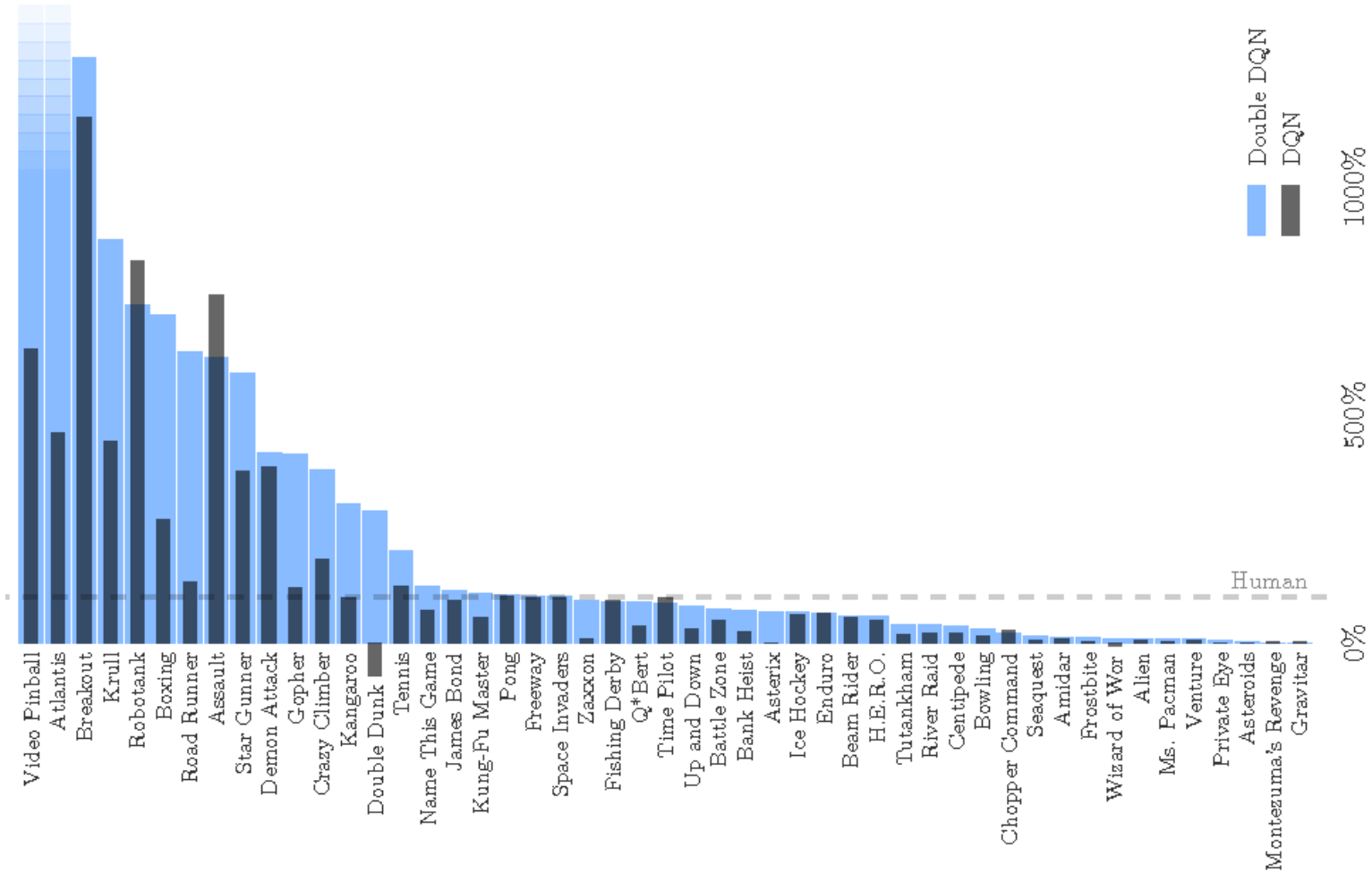
Autonomous driving

Walking robot

Play games


POWER
ON
OFF
COLOR
B-W
TV TYPE
VIDEO COMPUTER SYSTEM™
GAME SELECT
GAME RESET
ATARI

Human vs reinforcement learning



Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Thirtieth AAAI conference on artificial intelligence.

Reward

A **reward** R_t is a scalar feedback signal.

It indicates how is the agent performing at time t.

The agent's job is to maximize the **cumulative reward**.

Reward hypothesis

All goals can be expressed as the problem of maximizing some expected cumulative reward.

Examples of reward

- Autonomous driving:
  - $+$ for reaching the destination in time;
  - $-$ for crashing.
- Play chess:
  - $+$ for winning a match;
  - $-$ for loosing a match.
- Manage investment portfolio:
  - $+/-$ for each \$ earned/lost.
- Learn to walk:
  - $+$ for moving forward;
  - $-$ for falling.
- Play games:
  - $+/-$ for increasing/decreasing score.

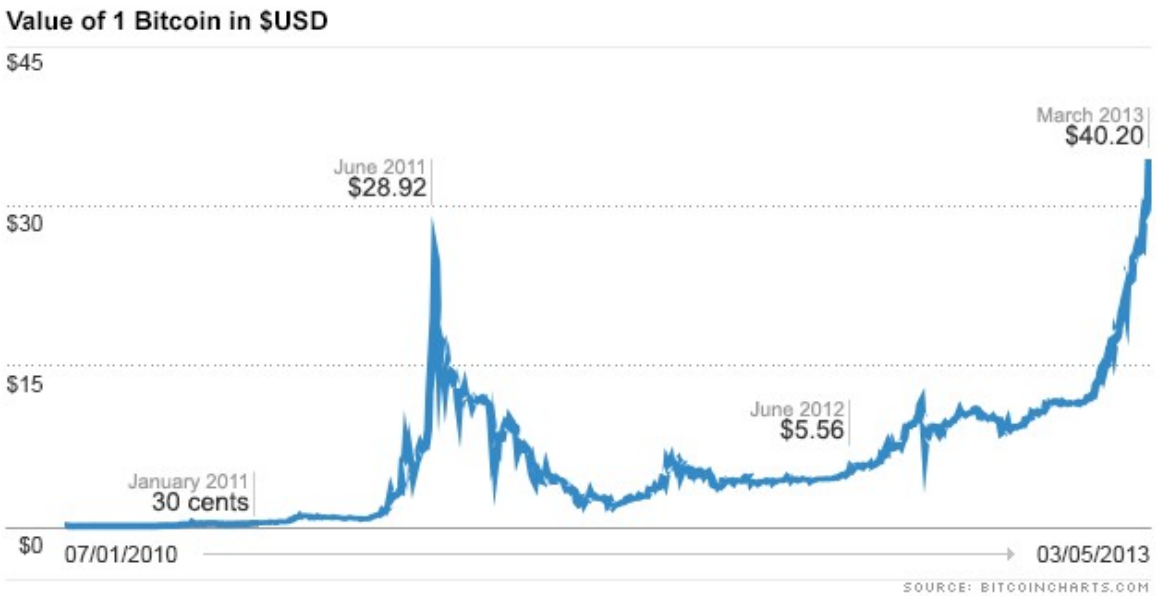
Sequential decisions

Goal: select actions to maximize **future cumulative reward**.

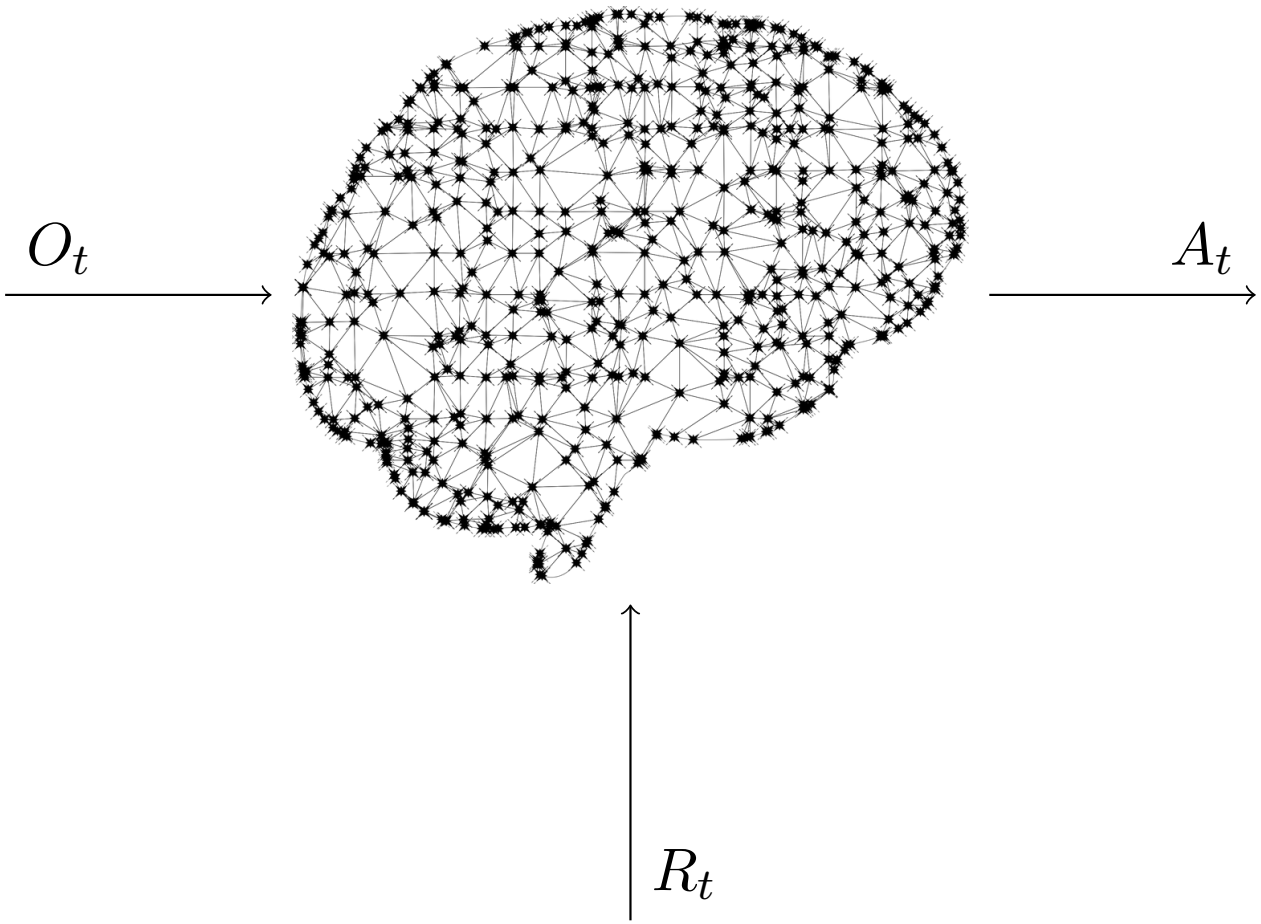
Actions may have long term consequences.

Rewards may be **delayed**.

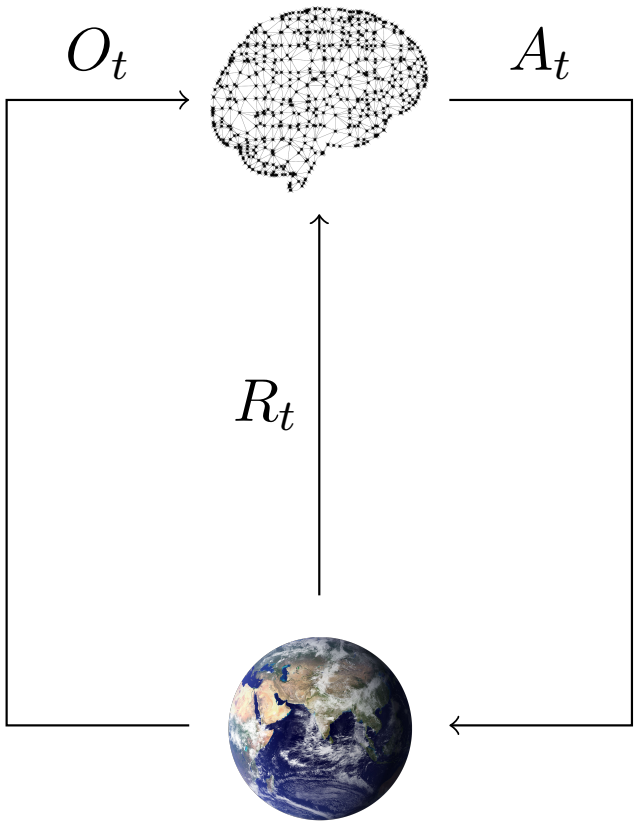
It may be better to *sacrifice immediate reward* to gain more after.



Agent



Environment



At each time t:

- The agent
  - receives observation O_t;
  - receives reward R_t;
  - takes action A_t.
- The environment
  - receives action A_t;
  - generates observation O_{t+1};
  - generates reward R_{t+1}.

History and state

The *history* is all that happened up to time t

$$H_t = \underbrace{O_1, R_1, \ldots, O_t, R_t}_{\text{Environment}}, \underbrace{A_1, \ldots, A_{t-1}}_{\text{Agent}}.$$

What happens next depends on

- the history H_t;
- the agent action A_t.

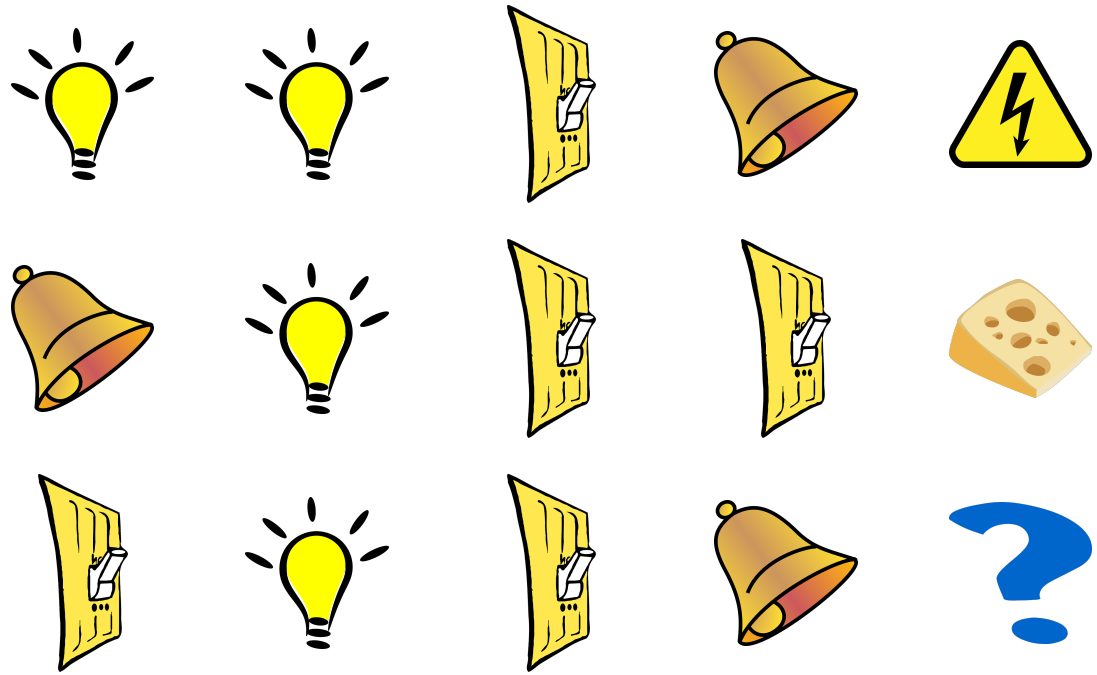
The *state* is the information used to select actions.

It is a function of history

$$S_t = f(H_t).$$

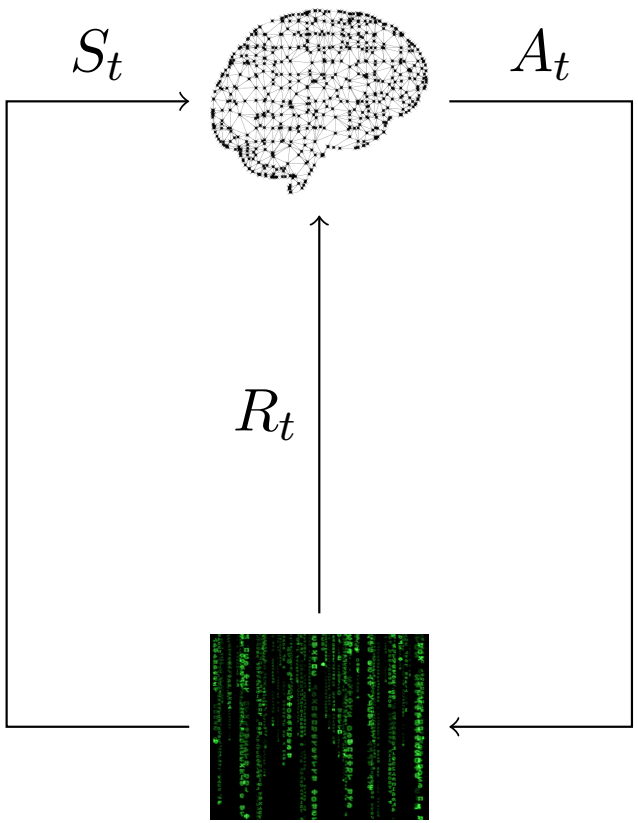
It must contain all the useful information about the history.

Mouse example



- What if S_t = last 3 items in sequence?
- What if S_t = counts for lights, bells and levers?
- What if S_t = complete sequence?

Markov state



The state S_t is Markov:

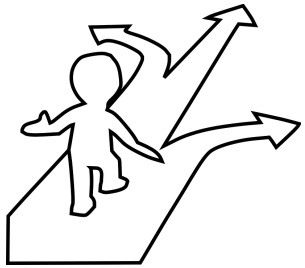
$$\mathbb{P}[S_{t+1}, R_{t+1}|H_t, A_t] = \mathbb{P}[S_{t+1}, R_{t+1}|S_t, A_t].$$

If the environment state is not accessible the agent has to construct its own representation:

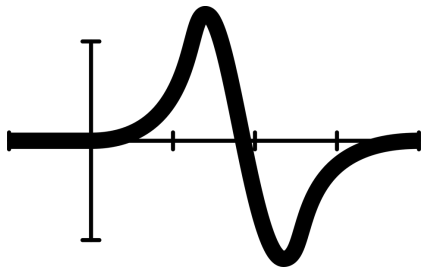
- use history H_t;
- beliefs on environment state;
- use a replica/estimator.

Ingredients of a reinforcement learning agent

Policy: rule used to take actions;



Value function: estimate of how good is each state;



Model: agent's belief on environment behavior.

Policy, value function and model

A *policy* is the agent's behavior.

It maps states to actions:

Deterministic	Stochastic
$a = \pi(s)$	$\pi(a\|s) = \mathbb{P}(a\|s).$

The *value function* predicts future reward.

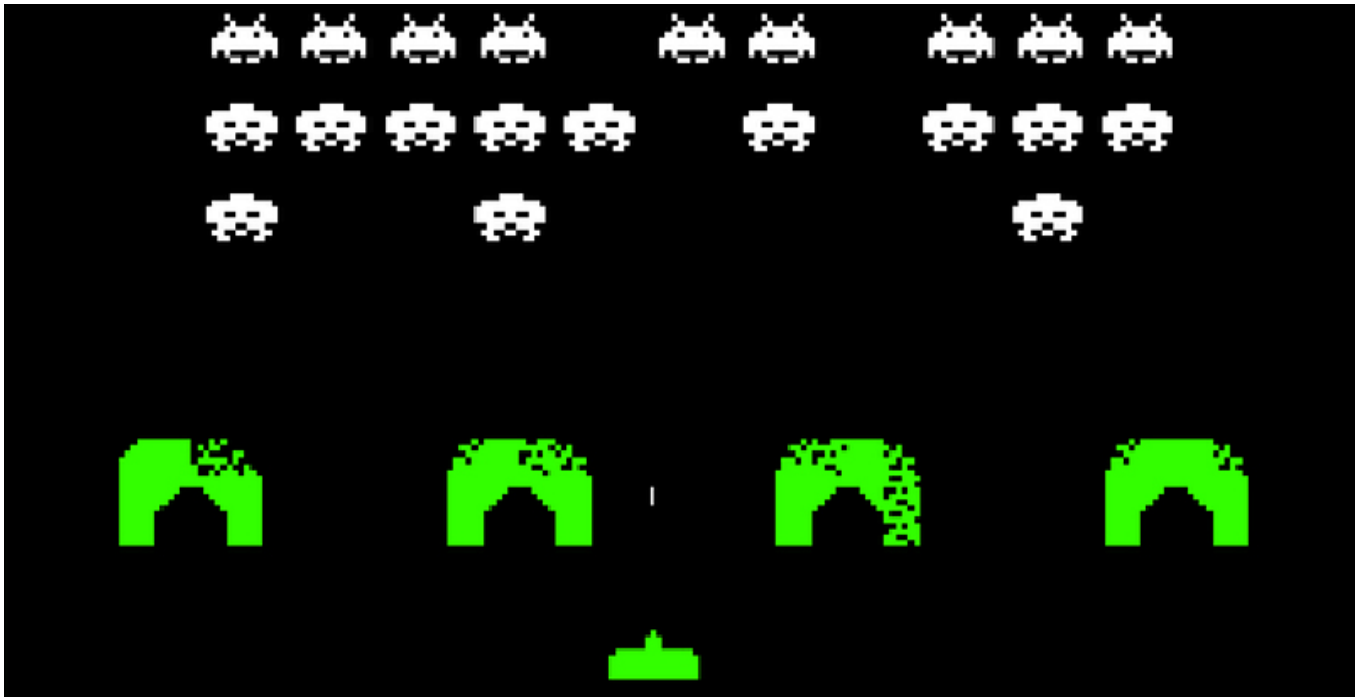
It is used to evaluate the goodness/badness of states,

$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |s\right].$$

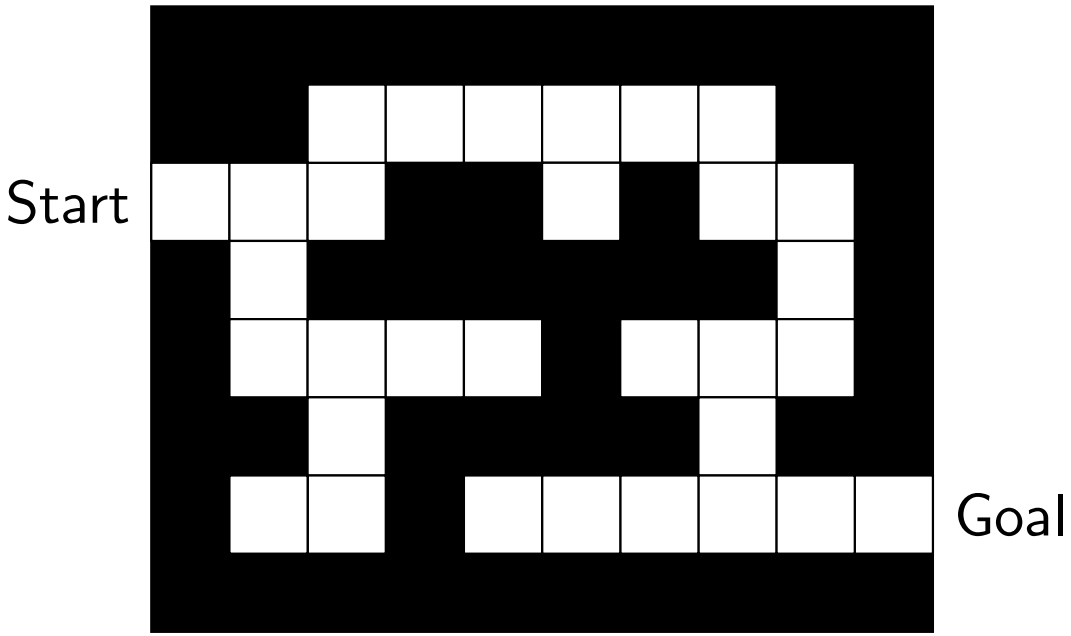
A *model* predicts what the environment will do next

$$\mathcal{P}^a_{ss'} = \mathbb{P}[s'|s,a],$$
$$\mathcal{R}^a_s = \mathbb{P}[r|s,a].$$

Example: space invaders



Maze

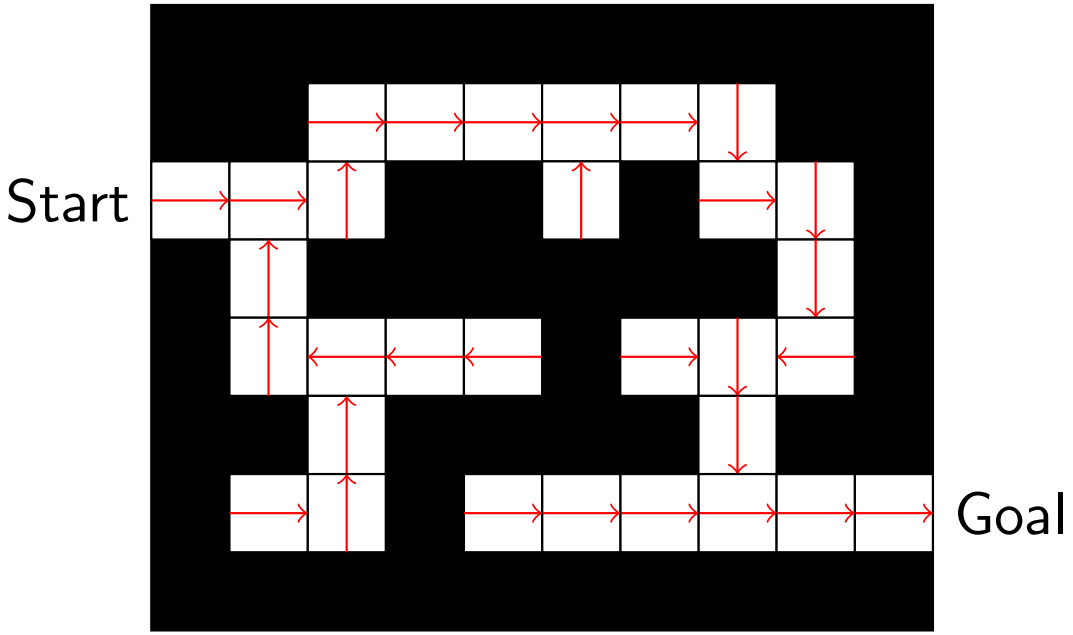


- Reward: $R_t = -1$;
- Actions: N, E, S, W;
- State: location.

Maze

Policy

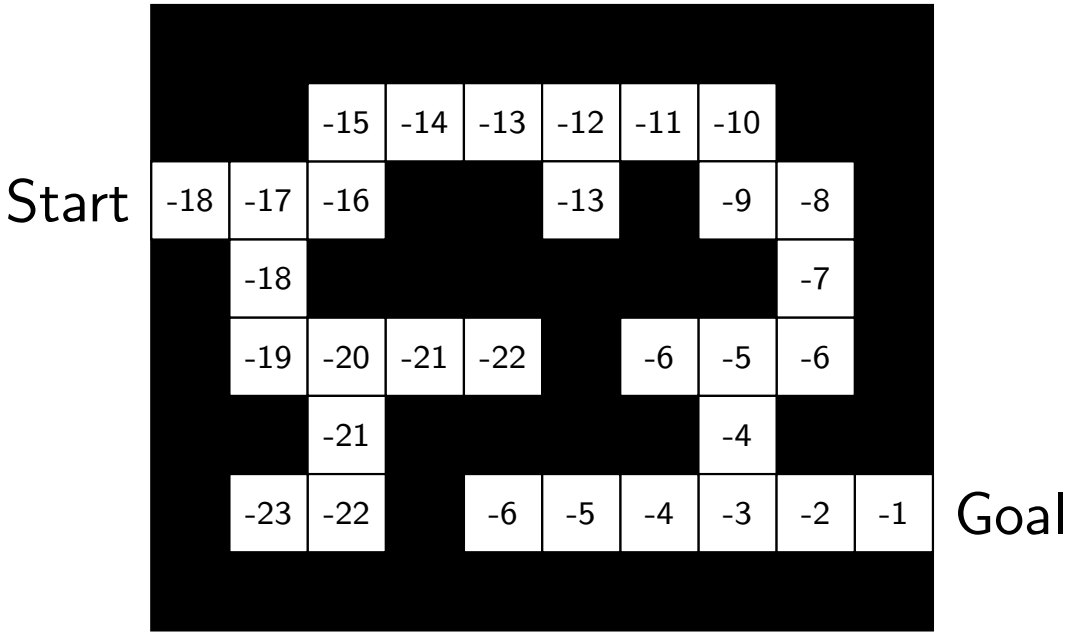
Deterministic policy $\pi(s)$ represented by arrows.



Maze

Value function

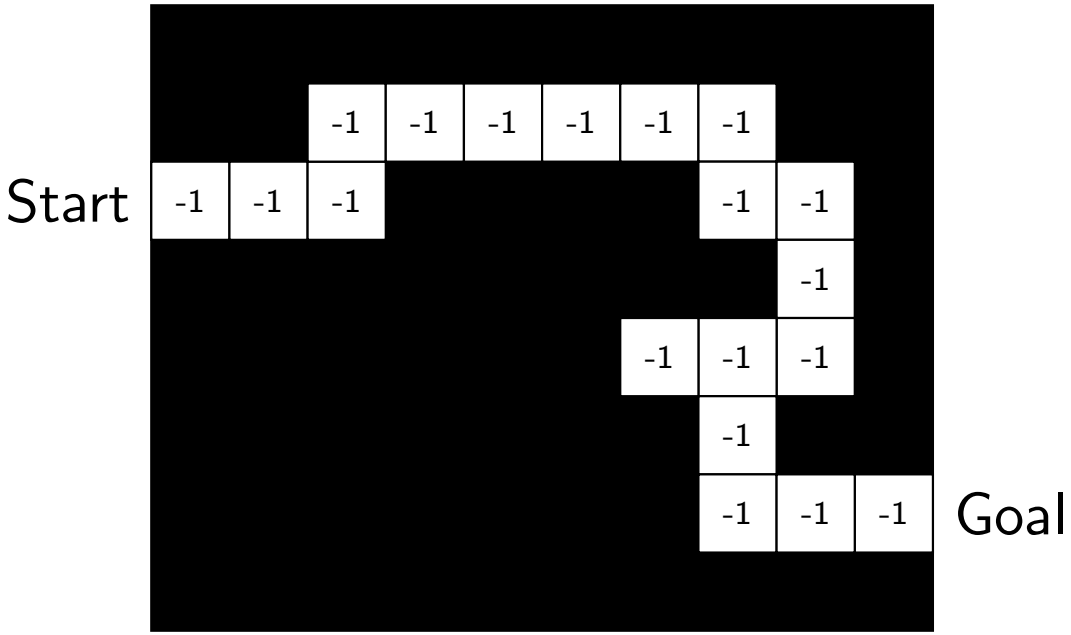
Numbers represents the value $v_\pi(s)$ for each state s.



Maze

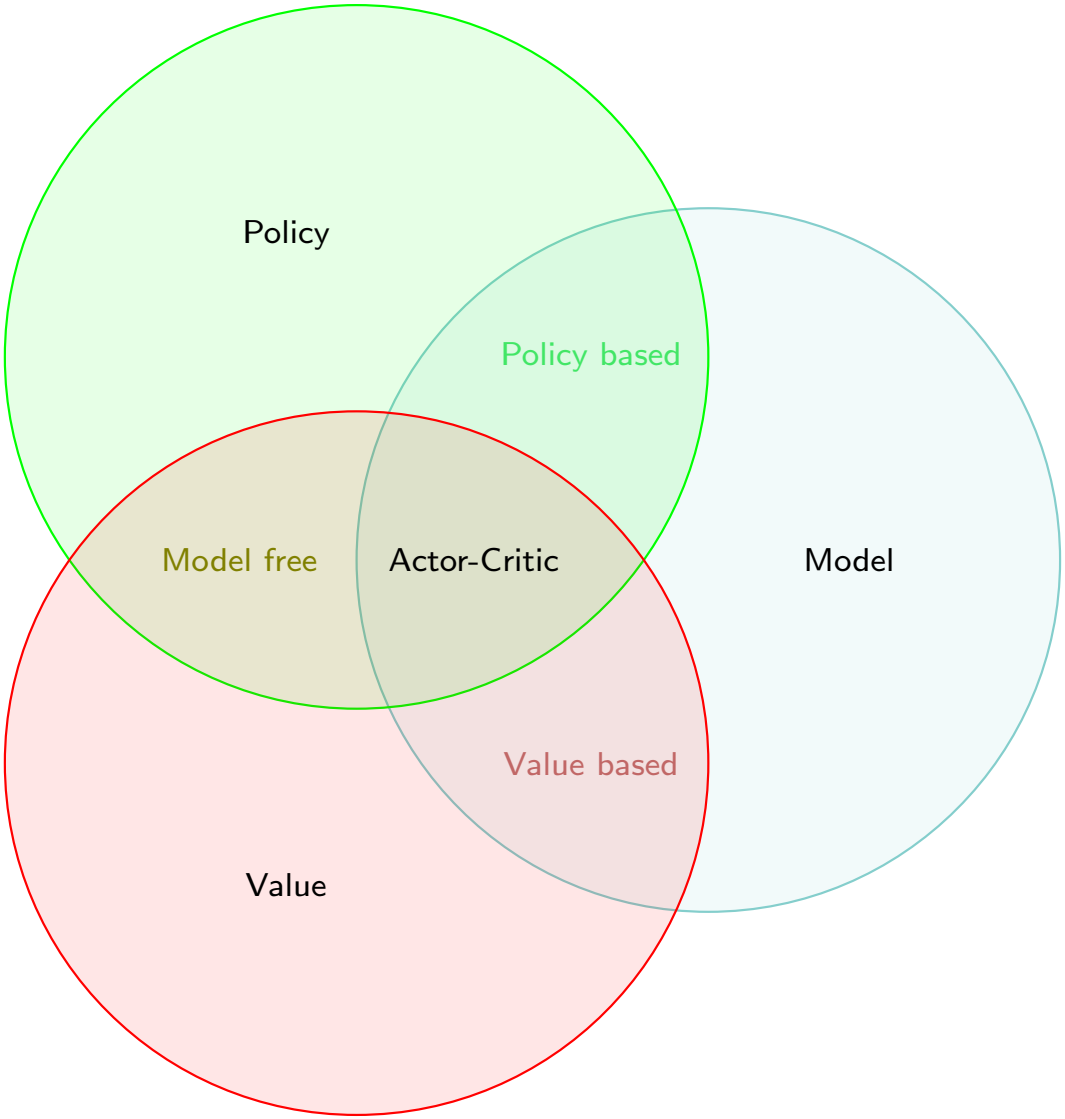
Model

The agent may have a representation of the environment



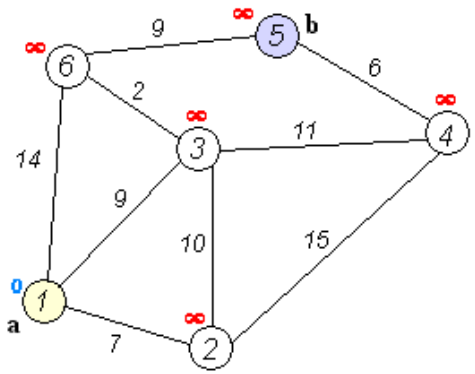
Layout represents transition probabilities.
Numbers represent immediate rewards.

Taxonomy of an agent



Learning vs planning

- Learning:
  - the environment is initially unknown;
  - the agent interacts with the environment;
  - the agent improves its policy.
- Planning:
  - a model of the environment is known;
  - the agent performs computations with its model;
  - the agent improves its policy.



Exploration and exploitation

Reinforcement learning is like trial-and-error learning.

The agent should discover a good policy without losing too much reward along the way, by balancing

exploration: find more information about the environment;

exploitation: exploits known information to maximize reward.

Usually ε-greedy policies are used:

- exploit with probability $1 - \varepsilon$;
- explore with probability ε.

Examples

- Restaurant selection:

  exploration: try a new restaurant;
  exploitation: go to your favorite restaurant.

- Online advertisement:

  exploration: try a different advert;
  exploitation: show the most successful advert.

- Gaming:

  exploration: play an experimental move;
  exploitation: play the move that you believe is the best.

Algorithms for learning and planning

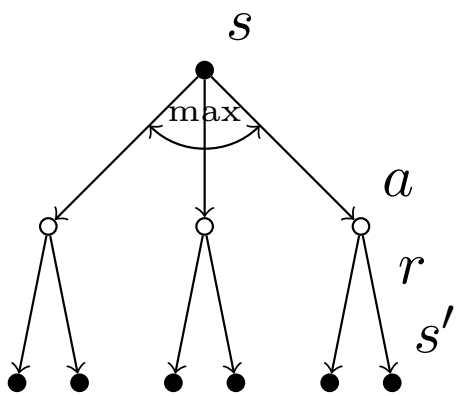
- Planning:
  - dynamic programming;
  - value iteration;
  - policy iteration;
  - ....
- Learning:
  - Q-learning;
  - ....

Dynamic programming

Define the *optimal state-value function*

$$v_\star(s) = \max_\pi v_\pi(s).$$

$v_\star(s)$ is the max we can get starting at s.



Bellman optimality principle

Tails of optimal trajectories are optimal:

$$v_\star(s) = \max_{a \in \mathcal{A}(s)} \sum_{r,s'} p(s', r|s, a)(r + \gamma v_\star(s')).$$

Policy iteration

- Initialization

  $V(s) \leftarrow \mathbb{R}$ and $\pi(s) \leftarrow \mathcal{A}(s)$ for all $s \in \mathcal{S}$

- Policy evaluation

  repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**
  $v \leftarrow V(s)$
  $V(s) \leftarrow \sum_{r,s'} p(s', r|s, \pi(s))(r + \gamma V(s'))$
  $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  until $\Delta < \theta$

- Policy Improvement

  for each $s \in \mathcal{S}$ **do**
  $\pi(s) \leftarrow \arg\max_a \sum_{r,s'} p(s', r|s, a)(r + \gamma V(s'))$

Value iteration

- Initialization

  $V(s) \leftarrow \mathbb{R}$ for all $s \in \mathcal{S}$

- Value estimation

  repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**
  $v \leftarrow V(s)$
  $V(s) \leftarrow \max_a \sum_{r,s'} p(s', r|s, a)(r + \gamma V(s'))$
  $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  until $\Delta < \theta$

- Policy extraction

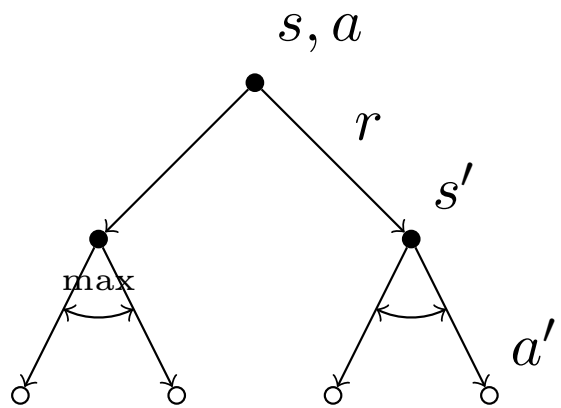
  for each $s \in \mathcal{S}$ **do**
  $\pi(s) \leftarrow \arg\max_a \sum_{r,s'} p(s', r|s, a)(r + \gamma V(s'))$

The Q-factor

Define the *optimal action-value function*

$$q_\star(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_{\star(S_{t+1})} | s, a\right].$$

$q_\star(s,a)$ is what we get taking action a and acting optimally afterward.



Bellman optimality principle

Can be rephrased in terms of $q_\star(s,a)$

$$q_\star(s,a) = \sum_{r,s'} p(s', r|s, a)\left(r + \gamma \max_{a' \in \mathcal{A}(s')} q_\star(s', a')\right),$$

since $v_\star(s) = \max_{a\in\mathcal{A}(s)} q(s,a)$.

Q-learning

- Initialization

  $q(s, a) \leftarrow \mathbb{R}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

- Learning

  repeat

  choose $a \in \mathcal{A}(s)$ using an ε-greedy policy

  take action a and observe r and s'

  $q(s, a) \leftarrow q(s, a) + \alpha \left(r + \gamma \max_{a'} q(s', a') - q(s, a) \right)$

  $s \leftarrow s'$

  until s is terminal

- Policy extraction

  for each $s \in \mathcal{S}$ **do**

  $\pi(s) \leftarrow \arg\max_a q(s, a)$

Conclusions and main challenges

Conclusions

- Reinforcement learning is a technique to take optimal actions balancing exploration and exploitation.
- If the state is measurable and finite, we have algorithms.

Challenges

- Infinite (uncountable) state space.
- Unmeasurable states.
- Continuous-time process.
- Exploration is not always possible.
- Conflicting goals (loss of reward hypothesis).
- Computational cost.